第1章 SystemVerilog，まずココに注目！

システム・レベル・モデリング以外の広範な機能を単一言語で網羅

高嶺美夫

ここではSystemVerilogの大枠を理解していただくために，現在のディジタルLSI設計が抱える課題と，SystemVerilogの導入によって解決される項目について紹介する．SystemVerilogは，従来の設計（回路）記述に加えて，アサーション・ベース検証や制約付きランダム・テスト生成，機能カバレッジといった手法に対応した検証記述を備えている．単なるVerilog HDLの改版（バージョン・アップ）ではなく，まったく新しい言語ととらえたほうがいいようだ．（編集部）

20年ぶりの新しい設計言語であるSystemVerilogが誕生しようとしています．“Verilog”という名を冠してはいますが，従来のVerilog HDLの延長線上では考えられない，新しい機能を備えた設計言語です．

本稿ではSystemVerilog誕生の背景と筆者が期待する点，実設計に適用する際に考慮するべき課題などについて紹介します．



図1　LSI設計の世界には“公用語”がない
最近，LSI設計の現場では，Verilog HDL以外にも多数の言語が使われている．単一の言語でできるだけ多くの範囲をカバーできるようにしようというのがSystemVerilog標準化のねらいである．

●システムLSI開発の現場に多数の記述言語がはんらん

Verilog HDLが設計に用いられるようになってから，約15年が経過しました．その間，ディジタルLSIの回路規模は約1,000倍に増大し，“システムLSI”や“SOC（system on a chip）”といったことばがあたりまえのように用いられるようになりました．これに伴って，設計の効率化を目的とした手法やEDAツールの開発が進められています．最近，とくに目につくのが，システム・レベル設計への移行と機能検証[注1]手法の進歩です．

システム・レベル設計とは，RTL（register transfer level）設計より前に，高い抽象度でLSI内部のアーキテクチャの検討を行ったり，携帯電話やディジタル・テレビなど，LSIを含むシステム全体を対象として最適化を行ったりする工程です．これは，開発の終盤で致命的な性能上の不良や機能不足などの不ぐあいが発生することを防止し，手戻りの少ない設計を行うことを目的として実施されます．また，機能検証は，現在のディジタルLSIの開発期間の半分，あるいはそれ以上を占める工程です．そのため，機能検証の効率化や高精度化が非常に重要な課題となっています．最近では，後述するアサーション・ベース検証や制約付きランダム・テスト生成などの手法が採用されています．

一方，ASIC（application specific integrated circuit）やシステムLSIを設計する際の主流の言語となっているVerilog HDLは，この間，ほとんど変化していません．正確に言うと，Verilog HDL 1995（Verilog 95）からVerilog HDL 2001（Verilog 2001）への改訂は行われたのですが，Verilog HDL 2001はEDAベンダによるツールのサポート

注1：おもにRTL（register transfer level）におけるHDLシミュレーションや等価性検証，デバッグなどの作業を指す．「機能検証」と対になることばは「タイミング検証（遅延解析）」．

が当初期待されたようには進まず，実際の設計現場で利用される機会も少ないのが現状です．

また，前述の機能検証手法を利用する場合，PSL（Property Specification Language）やe言語，Veraなど，それぞれ専用の言語を習得するという方法もあります．それぞれの言語には特有の長所があり，また，最近では言語仕様のオープン化も進んでいます．読者のみなさんの中にも，こうした言語を利用されている方がいらっしゃるかもしれません．ただし，現段階ではサポートしているEDAベンダの数が限られているという問題があります．ツールを言語ごとにそろえなければならないとなると，これは適用の大きな障害となってしまいます．

システム・レベル設計では，システムの機能を表現したモデルやアルゴリズムの記述が必要となります．これに対して，Verilog HDLではその要求に応えることが困難です．この分野ではC言語やC++，よりハードウェアの記述に適したSystemCなどが利用されています．

このように多数の言語が利用される状況の中で，設計言語を見直す必要性を，EDAベンダや半導体メーカなど，関係業界のだれもが感じていました（**図1**）．

● 単なる改版ではなく，まったく新しい言語

Verilog HDLの改訂は，IEEE（The Institute of Electrical and Electronics Engineers, Inc.）の専門委員会で5年ごとに行われることになっています．しかし，レビューや投票に時間がかかり，斬新な機能を早期に取り入れることが困難でした．そこでSystemVerilogの言語仕様の作成はEDAベンダが主体的に実施し，各社がいち早く対応することを約束しました．また，仕様策定をスピーディに行うことを目的として，業界主導の標準化団体Accelleraが言語仕様のとりまとめを行いました．

図2にSystemVerilogの位置づけを示します．Verilog HDL 2001の拡張仕様という形をとっていますが，Verilog HDLそのものも2005年版（IEEE P1364-2005）への改訂が同時に行われています．SystemVerilogはVerilog HDLの単なる改版（バージョン・アップ）ではなく，まったく新しい言語ととらえたほうがよいと思います．

しかしその一方で，新しい言語といいつつも，System Verilogに導入された機能の多くは，PSLなどによって有効性を実証済みの手法です．また前述のように，多くのEDAベンダが早期にサポートすることを表明しています．

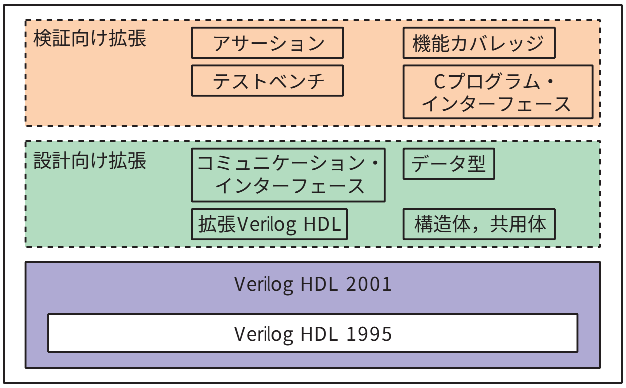


図2　SystemVerilogの位置づけ
もともとVerilog HDL 2001（IEEE Std 1364-2001）に対する拡張として策定されたが，大幅に機能が追加された結果，現在は設計と検証のための統合言語（HDVL：hardware design and verification language）となっている．

筆者がこの新しい言語に注目する理由がここにあります．

● システム・レベルやビヘイビア合成の比重が高まる

SystemVerilogについて説明する前に，ディジタルLSIの開発フローが今後どうなるかを考えてみましょう．

従来はRTL設計，シミュレーションを中心とした機能検証，論理合成というのがいわゆるフロントエンドの設計の流れでしたが，今後は**図3**に示すようなフローに変化していくと筆者は考えています．

こうした変化が生じる要因は二つあります．一つは，回路規模の増大に対応するためには，現状のRTL設計ではあまりにも粒度（設計者が取り扱う回路設計単位）が小さすぎることです．もう一つは，“システムLSI”と呼ばれることからもわかるように，システムとしての機能や性能などを評価・検証しておかないと，開発終盤になって致命的な不ぐあいが検出される可能性が高まっているためです．

この問題に対応するのが，システム・レベル設計（EDA業界では最近，ESL：electronic system levelということばが流行している）と，ビヘイビア合成（behavioral synthesis）または高位合成（high level synthesis）と呼ばれる技術です．

システム・レベル設計では，対象となるLSI，あるいはシステム全体の機能をモデル化し，高速なシミュレーションにより，性能評価やLSI内部のバス構成，メモリ構成などのアーキテクチャの最適化，仕様レベルの機能検証などを行います．最近の携帯電話のようにゲームをしながら電話を受けたり，音楽を聴きながらメールをしたりなどといった複雑なマルチタスク処理や，ディジタル・テレビにお

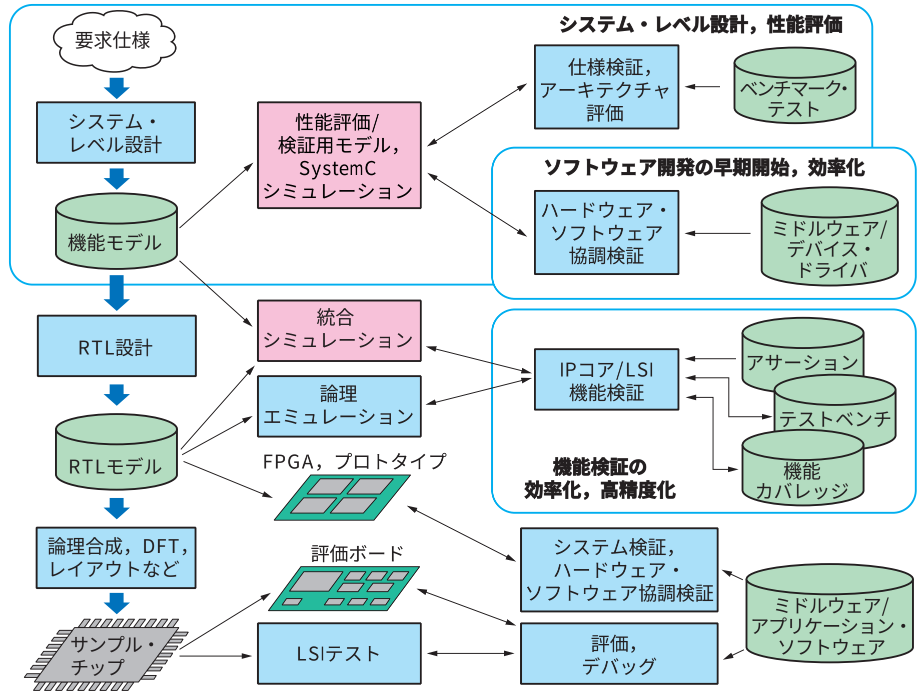


図3
システムLSIの開発フロー
システムLSIの大規模化やシステムの高機能化・高性能化に対応できる設計フローが求められている．

いて入力信号が乱れた場合でも映像が乱れないようにする処理など，RTLの機能検証では十分に評価できなかったり，RTL設計後の対処では手遅れとなってしまうような評価・検証作業を早期に実施します．

一方，ビヘイビア合成は，アルゴリズムやハードウェアの機能を，クロックを厳密に考慮せずに記述し，演算器などのリソースの割り付けやスケジューリングを含めてツールが最適化して論理を生成する技術です．

筆者ら（ルネサス テクノロジ）のような半導体メーカとしても，LSI（ハードウェア）を設計していれば十分という時代は終わっています．最近の組み込みシステムにおけるソフトウェアの開発規模は，LSIとは比べものにならない速さで増大しており，その開発負担を機器メーカだけに押し付けられる状況ではなくなっています．このため，とくに求められているのが早期にソフトウェア開発に着手できる環境の提供です．ここではC言語やC++，SystemCを用いた高抽象度のモデルの活用に注目が集まっています．

● 機能検証で三つの新しい手法が台頭

冒頭にも述べましたが，アサーション・ベース検証，制約付きランダム・テスト生成，機能カバレッジなど，機能検証の技術は大きく進歩しました．SystemVerilogは，これらに対応した構文や機能を備えています．図4に，これらの技術を活用した機能検証環境のイメージを示します．

アサーション・ベース検証は，設計の内部やインターフェース部にアサーションと呼ばれるチェッカ・モジュールを埋め込み，シミュレーション実行中の異常な動作を検出・報告するものです．この検出は自動的に実施されるため，シミュレーション結果の波形表示などを目視によって詳細に確認する必要がなくなります．また，原因となるバグに比較的近い箇所で異常を検出できるので，シミュレーション結果の確認と異常検出時の原因追及が容易になります．

制約付きランダム・テスト生成は，テスト項目（テスト・シナリオ）に対応するシミュレーション・パターンを作成する際に，乱数を用いてパターンを生成するものです．乱数を用いることにより，（人間が作成できないような）多様な条件について検証することが容易になります．ただし，何も考えずに乱数でパターンを生成すると，非現実的で意味のない条件を生成してしまいます．そこで，有効なパターンのみを生成するように制約を与えることが重要になります．例えば，メモリ・アクセス時のアドレスを生成する場合に下位3ビットをつねにゼロとしたり，複数の信号の組み合わせについてある条件を生成する確率を指定するといった制約を与えることができます．ランダム・テスト生成によるパターンを印加して得られたシミュレーション結果は，リファレンス・モデルによって生成された期待値と比較されます．また，上述のアサーションを用いて異常動作を検出する際にも，このパターンが利用されます．

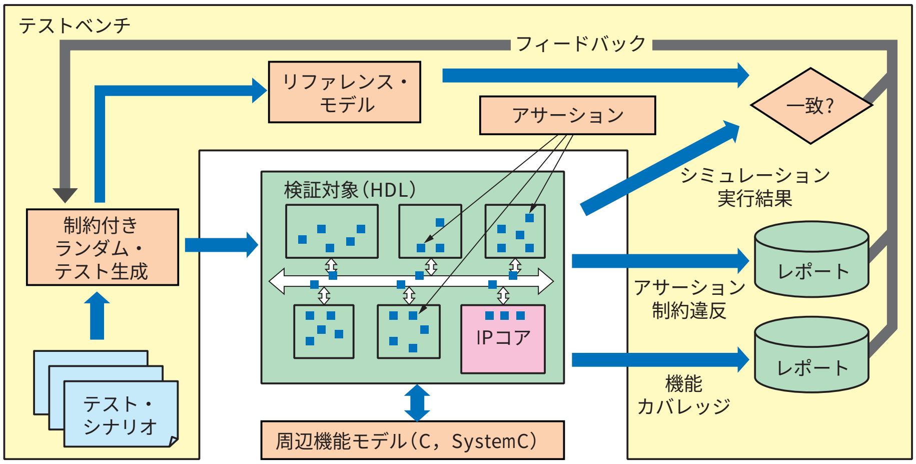


図4
機能検証環境
検証の効率化・高精度化のため，テストベンチはますます高機能になってきた．

カバレッジは，シミュレーションによって意図する条件がどの程度チェックされたかを数値化し，検証の進ちょく度を表すものです．従来はHDLコードにおいて，各ステートメントが実行されたかとか，条件式が判定されたかなどをカバレッジの指標として用いていました(いわゆるコード・カバレッジ)．最近ではそれらに加えて，特定の動作(トランザクション)が実行されたかどうかを指標とする「機能カバレッジ」が，より実際的にシミュレーションの網羅性を示すものとして使用されています．さらに，カバレッジの評価結果をアサーション・ベース検証の結果などとともにランダム・テスト生成の制約に反映させる(例えば，網羅されなかったシミュレーション・パターンを優先的に生成させる)ことにより，より効率の良い検証を実施する手法も利用されています．

もう一つ，検証において考えておかなければならないことは，検証に関する資産(IPコアなど)の活用です．例えば，IPコアに検証のための機能，すなわちアサーションや，ランダム・テスト生成を行うトランザクタ，カバレッジの条件などをあらかじめ埋め込んでおくのです．このようなIPコアを利用することにより，LSI全体や関連するブロックの検証を効率良く行えるようになります．

● 乱立する検証言語の統一を目指す

システム・レベル設計やハードウェア・ソフトウェア協調検証に対して，C言語やC++，SystemCを利用することが珍しくなくなっています．これは，もともとシステムの機能(アルゴリズム)がほとんどソフトウェア・プログラムとして記述されていること，モデリング言語としてC++が有効な性質(オブジェクト指向)を備えていること，加えてC++の特徴とハードウェア記述に適した特徴を兼ね備えたSystemCが登場したことなどが背景にあります．

このような機能をハードウェア記述言語に付加することも可能ですが，一方，モデリングと設計は別物と考えて，システム・レベル言語とハードウェア記述言語を使い分けるという選択もあります．この場合，両者のモデルを接続する手段を用意し，接続に対する親和性を確保することが重要となります．

機能検証については，前述したようにPSL，e言語，Veraなどの専用言語を使い分け，さらにそれらに合わせて使用するツールを選択するという状況にあります．ひとりのエンジニアがすべての言語に精通することは現実的ではなく，またプロジェクトによっては，EDAツールがサポートする言語の問題から，利用可能な手法が制約されてしまうといった問題が起きています．このため，とくに機能検証については言語が統一されること，少なくとも各EDAベンダのツールが必要な言語を同じレベルでサポートすることが望まれています．

● 言語上の改良・追加機能は四つ

以上，上流の設計と機能検証の問題，および今後の設計フローについて説明しました．次に，これらと関連するSystemVerilogの特徴と，実際に適用する際の課題について述べます．

表1に特徴を示します．ここではSystemVerilogを利用する立場から，その期待できる効果を述べます．

1) 情報を効率良く記述する記法を導入

SystemVerilogの設計言語としての第1の特徴は，設計(回路)をコンパクトに記述できることです．インターフェ

1

表1
SystemVerilogの特徴

分　類	特　徴	関連する構文
簡潔な設計（回路）記述	●コード量を1/2～1/5に削減可能 ●誤りを少なくし，保守を容易に	インターフェース，構造体，共用体，列挙型，外部定義型，暗黙のポート接続
記述のあいまいさの排除	●記述者の設計意図を明示	always_comb/always_latch/always_ff, unique/priority
機能検証の効率化	●アサーション・ベース検証，制約付きランダム・テスト・パターンによる検証，機能カバレッジの取得を容易に ●設計者と検証エンジニアのコミュニケーションを容易に ●検証容易なIPコアを提供	SystemVerilogアサーション（SVA），制約付きランダム・テスト生成，機能カバレッジ，インターフェース，クロッキング・ブロック
高抽象化記述	●システム・レベル・モデルとの接続を容易に ●ツールの処理効率を向上	Direct Programming Interface（DPI），2値型，クラス

ースの記述を階層化したり，C言語と同様の構造体，共用体などを用いることにより，記述量を数分の1程度とすることができます．これは単に記述工数を削減するだけでなく，可読性を向上させ，誤りを少なくします．また，保守を容易にして再利用性を高めることにつながります．

SystemVerilogでは，設計抽象度を引き上げる，すなわち情報量を減らす（情報を間引く）のではなく，同じ情報を効率良く記述する記法を多数導入している点が大きな特徴です．

2）記述のあいまいさを排除

Verilog HDLでは，RTLシミュレーション時に正しく動作していたものが，論理合成後に正しく動作しなくなるという問題がしばしば発生していました．これは多くの場合，設計者の意図と異なる回路が合成される（例えば，組み合わせ回路で実現することを期待している箇所にラッチが生成される）ことにより発生していました．

SystemVerilogでは，組み合わせ回路やラッチ，フリップフロップの生成，およびcase文などの判定について，設計者の意図を明確に表現できる記法が用意されています．これにより，論理合成後の回路の動作を保証できるようになります．新しい記法の多くは，従来，論理合成ツールに対する指示（プラグマ）として記述していた情報を，統一された言語仕様の中で記述できるようにしたものです．ツール間の取り扱いや解釈の差を解消し，設計者の意図と異なる回路が生成される状況を避けることができます．

3）注目されている機能検証手法に対応

SystemVerilogの最大の特徴とも言える点が，最近はやりの検証手法を取り込んだことです．前述のアサーション・ベース検証，制約付きランダム・テスト生成，機能カバレッジなどを，一つの言語の中で利用可能となります．このほかにも，テストベンチを簡潔に記述するための機能などを備えています．

一つの言語の中でこれらの手法を利用できることにより，設計資産としての再利用性が向上し，また後述するように設計者と検証エンジニアの間のコミュニケーションが円滑になるといった効果が期待できます．

4）高抽象化記述

SystemVerilogは，記述の抽象度を上げる記法を用意しており，同時にC言語などで記述した機能モデルとの接続が容易になるという特徴もあります．

前者については，C言語と同じ2値型が広範囲に導入されています．これにより，とくにシミュレーションの高速化が期待できます．

後者については，DPI（Direct Programming Interface）と呼ばれる新しいインターフェースが用意されました．Verilog HDLでは，Cプログラムとのインターフェースとして PLIが提供されていますが，PLIには，記述が煩雑で，シミュレーション実行時の速度オーバヘッドが大きいといった問題があります．DPIはこうした問題を解消します．Cモデルとの接続が容易になり，シミュレーション速度が向上することを期待できます．

● 設計者と検証エンジニアの言語を統一

SystemVerilogを利用することにより，システム・レベルのモデリングを除くほとんどの設計工程が一つの言語でカバーできることになります注2．これは設計チームにとって大きなメリットになります．

まず，言語習得については，移行期こそ習得のための相応の努力が必要となりますが，複数の言語を習得するより

注2：その名称ゆえに誤解されがちだが，SystemVerilogではシステム・レベル・モデリングのための機能はそれほど強化されていない．一方，SystemCはシステム・レベル・モデリング向きの言語と言える．

ははるかに楽になると期待されます．さまつなことですが，セミコロンなどの記号の使いかた一つとってみても，言語ごとに異なるスタイルを覚えることは非常に煩雑に感じます(そのうえ，これは設計業務の本質とは関係ない)．

プロジェクトを計画・管理する立場から言うと，利用できるツールの選択肢が増え，自由度が増すことにより，計画・管理が行いやすくなると期待できます．もっともこれは，課題として後述するように，EDAベンダのきちんとしたサポートが必要条件となります．また，言語仕様として定義されたことにより，ツールに依存する制約が減少し，設計資産の活用がより円滑になると期待されます．

さらに，設計者と検証エンジニアが同じ言語を使用することにより，コミュニケーションが円滑になると期待できます．日本ではそれほど一般的ではありませんが，米国などの大規模な設計プロジェクトでは，与えられた仕様から回路を設計する設計者と，その回路を検証する検証エンジニアが分かれています．両者の職務を明確に区別し，異なる視点でクロスチェックを行うことにより，検証漏れが生じたり，市場に不良品が流れてしまうリスクを低減できると考えられています．

従来，設計者はもっぱらHDLのみを使用し，両者のインターフェースとなる仕様は不完全なフローチャートや自然言語で与えられることがほとんどでした．SystemVerilogを用いると，仕様をアサーションとして記述したり，設計者が設計の意図を明確に表現しやすくなります．これによりプロジェクト全体の設計生産性が向上することを期待できます．

● 設計記述についてはチェッカの整備が必要

SystemVerilogを導入・利用する際には，いくつか気をつけなければならない項目があります．

まず，現在Verilog HDLを入力としている各種EDAツールがSystemVerilogの必要な機能を完全にサポートするまでは，おもに設計(回路)記述において問題が生じる可能性があります．例えばシミュレータや論理合成ツールがSystemVerilogのある機能をサポートしていたとしても，そのほかの設計ツール(社内製ツールを含めて)がサポートしていないと，下流の設計工程で不ぐあいが生じる可能性があります．このように問題のある機能は，設計記述中の構文や構造を静的に解析する"チェッカ"と呼ばれるツールによって完全に排除する必要があります．つまり，社内で利用するツール全般のSystemVerilogのサポート状況を調査し，チェッカを整備することが必要となります．

ちなみに検証記述については，再利用性や移植性について留意すべき点はあるものの，上述のような設計上の致命的な問題はほとんどありません．SystemVerilogを使えるところから使っていく，という姿勢でよいと考えています．

もう一つ注意しなければならないのは，言語仕様に関するサポート状況を調べるだけでは，実用上問題がないのかどうかわからない機能が存在する点です．これはツール間の性能差と言語仕様の解釈の違いという形で現れる可能性があります．現状でも，例えば論理合成において，ツール間で能力差が顕著に現れます(生成される回路に優劣がある)．SystemVerilogでは，各ベンダとも新しい言語をサポートするということで，当初は実装の違いにより無視できない能力差が現れることが予想されます．代表的なものは，制約付きランダム・テスト生成です．ツール側のアルゴリズムによっては，パターン生成の効率に大きな差が生じる可能性があります．

言語仕様の解釈については，現時点で具体的にどのような問題が起こるのかを例示することは困難です．実際に言語仕様を策定しているのは限られた人数のエンジニアであるのに対して，今後，多数のEDAベンダやEDAユーザが評価にかかわってきます．そうなると，言語仕様上のあいまいな点が浮き彫りになってくる可能性があります．

＊　　＊　　＊

Verilog HDL 2001が登場したとき，「使ってみたい」と思いながら，現実にはそうならず，失望したことがありました．仕様の策定にEDAベンダの代表が含まれていなかったことがその一因と言われています．一方，今回のSystemVerilogについては，大手EDAベンダの代表が主導で仕様策定を推進してきた経緯があります．すべての機能が早期に利用可能となることを期待したいと思います．

たかみね・よしお
(株)ルネサス テクノロジ

<著者プロフィール>
高嶺美夫．日立製作所入社以来，コンピュータや半導体設計向けの高速シミュレータ，機能検証，高位合成，DFT，システム・レベル設計技術などの研究開発に従事．JEITA EDA技術専門委員会SystemVerilogタスクグループのメンバとして言語標準化に携わっている．訳書に「SystemVerilogによるLSI設計」(2005年1月，丸善刊)がある．